

FSA Integration Partner Program
United States Department of Education
Office of Federal Student Aid



FSA Web Security Standards

Version 1.0

November 26, 2003

Document Revision History

Version Number	Date	Author	Revisions Made
1.0	November 26, 2003	Jesse Bowen	

Table of Contents

1	EXECUTIVE SUMMARY.....	5
2	INTRODUCTION.....	6
2.1	BACKGROUND.....	7
2.1.1	<i>Web and Internet Security Threats</i>	<i>7</i>
2.1.2	<i>Relationship to FSA Security and Privacy Policy.....</i>	<i>8</i>
2.2	ORGANIZATION OF THIS DOCUMENT	8
2.3	DEFINITIONS	9
3	SCOPE AND APPLICABILITY OF STANDARD	10
3.1	SCOPE OF WEB SECURITY STANDARD.....	10
3.2	APPLICABILITY OF STANDARD	10
3.3	INTEGRATION WITH FSA SOLUTION LIFE CYCLE	10
3.4	USE OF STANDARDS AND GUIDELINES	10
4	FSA WEB SECURITY STANDARDS AND GUIDELINES	12
4.1	SECURITY OF NETWORK AND INFRASTRUCTURE FOR WEB COMPONENTS	14
4.1.1	<i>Background.....</i>	<i>14</i>
4.1.2	<i>Standard</i>	<i>14</i>
4.1.3	<i>Guidelines.....</i>	<i>14</i>
4.1.4	<i>References.....</i>	<i>18</i>
4.2	SECURITY OF WEB SERVERS	19
4.2.1	<i>Background.....</i>	<i>19</i>
4.2.2	<i>Standard</i>	<i>19</i>
4.2.3	<i>Guidelines.....</i>	<i>19</i>
4.2.4	<i>Server-Server Trust in Web System Design</i>	<i>24</i>
4.2.5	<i>References.....</i>	<i>26</i>
4.3	SECURITY OF WEB APPLICATIONS	27
4.3.1	<i>Background.....</i>	<i>27</i>
4.3.2	<i>Standard</i>	<i>27</i>
4.3.3	<i>Guidelines.....</i>	<i>27</i>
4.3.4	<i>References.....</i>	<i>34</i>
4.4	WEB SERVICES SECURITY STANDARDS.....	35
4.4.1	<i>Background.....</i>	<i>35</i>
4.4.2	<i>Standard</i>	<i>36</i>
4.4.3	<i>Guidelines.....</i>	<i>37</i>
4.4.4	<i>References.....</i>	<i>39</i>
4.5	ENCRYPTION FOR WEB COMPONENTS	40
4.5.1	<i>Background.....</i>	<i>40</i>
4.5.2	<i>Standard</i>	<i>41</i>
4.5.3	<i>Guidelines.....</i>	<i>41</i>
4.5.4	<i>References.....</i>	<i>43</i>
5	IMPLEMENTATION APPROACH.....	45
5.1	REVIEW OF STANDARDS	45
5.2	PUBLICATION OF STANDARDS	45
5.3	ENFORCEMENT AND MONITORING	46
	APPENDIX A: LIST OF TCP/IP PORTS TO CONSIDER BLOCKING.....	47
	APPENDIX B: REFERENCES.....	49
	APPENDIX C: ADDITIONAL SECURITY DEFINITIONS.....	51
	APPENDIX D: FSA SECURITY AND PRIVACY TECHNICAL ARCHITECTURE.....	53

List of Figures

Figure 1- Security Mechanisms to Protect Server-to-Server Communications.....	25
Figure 2 - Business Benefits of Web Services Security.....	35
Figure 3 – Technical Benefits of Web services security	36
Figure 4 – Overview of Standards under Development	38
Figure 5 - Recommended uses of encryption algorithms approved for federal information systems.	43
Figure 6 - List of TCP/IP Ports Commonly Associated with Network Attacks...	48
Figure 7 – FSA Security and Privacy Technical Architecture.....	53

1 Executive Summary

Web applications offer FSA several benefits for providing services to borrowers and trading partners. Many FSA systems already have Web-based components, and more are planned for the future. Web-based architecture is attractive because of its high compatibility with user client software and ability to interface with existing FSA legacy systems. Unfortunately, the use of Web technology also poses a variety of security challenges specific to Web technology and use of the Internet. Due to its accessibility to potential external attackers, Web-based systems face a variety of threats to the confidentiality, integrity, and availability of FSA data. These threats are different in nature or magnitude compared to systems only accessible internally to FSA.

This document proposes technology and operational standards for FSA to adopt to provide appropriate protection for FSA systems and data that are exposed externally via Web applications communicating over the Internet. Specific FSA Web Security Standards and Guidelines standards were developed to address the following common design elements of Web applications:

- Network and Infrastructure Components
- Web Servers
- Web Applications
- Web Services Security Standards
- Encryption for Web Components

Information on each Web standards area is organized by providing information on its Background, Standard, Guidelines, and References.

The standards and guidelines for development, operation, and testing of Web applications drafted in this document must be interpreted in light of specific FSA business objectives, technical constraints, and operational requirements. A number of additional implementation steps will be required to ensure the standards are adequately communicated and used by the FSA employees and contractors responsible for designing, developing, and deploying Web applications. The approach outlined in this document for adopting web standards includes the following steps:

- *Review of Standards* to validate that they meet FSA requirements for protecting Web applications.
- *Publication of Standards* to communicate the standards to a wide audience and improve their accessibility and visibility.
- *Enforcement and Monitoring* including incorporation into the FSA SLC and periodic review to maintain changes in Web technologies, threats, and recommended security controls for Federal information systems.

2 Introduction

Web applications offer a variety of benefits to help FSA provide effective services to borrowers and trading partners. Several FSA systems already have Web-based components, and more are planned for the future. Web-based architecture is attractive because of its high compatibility with user client software and ability to interface with existing FSA legacy systems. But the use of Web technology also poses a variety of security challenges specific to Web technology and use of the Internet. Because of its accessibility to external potential attackers, Web-based systems face a variety of threats to the confidentiality, integrity, and availability of FSA data that are different in nature or magnitude compared to systems only accessible internally to FSA. This document proposes technology and operational standards FSA plans to adopt to provide appropriate protection for FSA systems and data that are exposed externally via Web applications communicating over the Internet.

The FSA Information Technology Security and Privacy Policy states that:

“If the public accesses an FSA system, FSA must develop and implement security controls to protect the integrity of the application and the confidence of the public. Each FSA Web page must have a designated author or administrator who is responsible for ensuring Web page security. If a server contains information protected by the Privacy Act, it must not be accessible without proper authorization. Additionally, the Website should provide notice that it contains Privacy Act information, and give notice of the consequences of unauthorized disclosure. Users wishing to access internal FSA systems via the Internet must be authenticated.”

To support this policy, specific standards have been developed to address the following common design elements of Web applications:

- Network and Infrastructure Components
- Web Servers
- Web Applications
- Web Services Security Standards
- Encryption for Web Components

This document will also provide recommendations on how these Web security standards should be reviewed, implemented, monitored, and enforced. The proposed Web security standards is primarily based on federal information security recommendations and guidelines, as described in the references sections for each standard. Where appropriate, guidelines and suggestions for specific Web security issues, tools, and testing procedures are presented from private sector sources to augment the recommendations.

2.1 Background

2.1.1 Web and Internet Security Threats

Use of the Internet and Web applications to provide services to FSA customers and business partners is growing rapidly. Web technologies offer opportunities to deliver services to a variety of users with minimal deployment of client software, while allowing relatively rapid integration with both existing and new FSA systems. Most future FSA systems will likely provide a Web interface for at least some of their functionality. Current FSA planning address strategies for even broader deployment of functionality via Web services, which offer the ability to deploy business functionality as discrete functions that are flexible and quickly adapted to business needs.

Given the continuing growth of Web technologies, it is important for FSA to develop and communicate security standards for the Web environment. Basic security principles and objectives for Web applications are no different than those for other technologies. But some types of threats, vulnerabilities, and risks are specific to the Internet and Web applications. Likewise, the FSA Certification and Accreditation process should in theory lead to appropriate planning, reviewing, and testing of Web applications. Yet there are specific testing procedures and other considerations that apply to Web applications and technologies. The FSA Web security standards and guidelines will help identify vulnerabilities and threats unique to Web applications that should be considered during review of these systems.

Although use of the Internet and Web applications offer several advantages over more traditional architectures, they also pose new types of security threats. Reasons for the new types of security threats introduced by Web applications include:

- Increased access by external users – Web applications are designed to facilitate access by new populations of users that are not subject to the same methods of identification available for internal users.
- Connections to the Internet – Web applications intended for external use must be connected to the Internet, allowing potential access to an extremely large population of unauthorized users.
- Changing standards and protocols – rapidly changing Internet standards and protocols that define interactions between Web components introduce new forms of security risk that may not be immediately understood.
- New Web hardware and software – frequent introduction of new Web software and hardware technologies provide limited time to fully investigate and validate security control mechanisms.
- Complexity of Web architectures – Web applications are rarely static, standalone systems. Increasingly, Web applications provide dynamic content with multi-tier architecture designs that provide external access to core internal systems that can display or alter sensitive internal or customer data.
- Web ‘hacking’ culture – A large pool of attackers have developed techniques and tools that are widely available to probe, subvert, or damage Web applications for profit, malicious purposes, or merely entertainment.

FSA faces several unique challenges in protecting its Web applications. Because FSA financial aid programs provide services to a very large population of borrowers and trading partners, its Web applications will be subjected to the scrutiny of millions of users. FSA systems administer large financial portfolios, and store sensitive personal and financial information about customers. As a result, FSA Web applications may be high-profile targets for attackers. Yet general experience of similar organizations has demonstrated that effective implementation of good practices for design, development, and operation of Web sites can significantly mitigate the risks inherent to use of the Internet and Web applications.

Development and implementation of Web security standards will benefit FSA, its developers, and contractors responsible for designing and deploying FSA Web applications. Web security standards will provide guidance to establish a common set of security concerns that should be addressed by Web application designs.

2.1.2 Relationship to FSA Security and Privacy Policy

FSA has developed an Information Technology Security and Privacy Policy. This policy provides overall guidance to protect FSA systems and data. FSA has also developed a Security and Privacy Architecture to define technical design elements for protecting FSA systems. The Security and Privacy Architecture¹, as well as a gap analysis conducted during an earlier phase of this task order², identified a need to develop Web security standards. This Web security standard provides additional guidance specific to the design, development, and operation of FSA Web applications and Web application components. Standards and guidelines presented in this standard are intended to be consistent with both the overall security and privacy guidance documented in FSA policy, and with the FSA Security and Privacy Architecture.

2.2 Organization of This Document

This deliverable was created as part of Task Order 120 and provides recommendations on reviewing, implementing, monitoring, and enforcing Web Security standards at FSA.

The remainder of this document consists of three major sections, described briefly below:

- Section 3 – *Scope and Applicability of Standard* provides an overview of the scope and applicability of the Web standards for FSA.
- Section 4 – *FSA Web Security Standards and Guidelines* provide guidance for the development and deployment of FSA Web applications including network and infrastructure security for Web components, security for Web servers and applications, Web security standards, and encryption.

¹ Deliverable 124.1.2 – Final Security and Privacy Architecture report, and Deliverable 124.1.3 – Security and Privacy Architecture Specification.

² Deliverable 120.2.3 – Security Architecture Status Report – Aug-Sep 2003.

- Section 5 – *Implementation Approach* describes the contractor and developer support necessary to deploy the FSA Security and Privacy Architecture.

2.3 Definitions

Below are definitions of security and networking terms important to the discussion of Web security standards and guidelines. Additional definitions of more general security terms and concepts are provided in Appendix C.

Cryptography: The science and its application concerning the principles, means, and methods for rendering plain text unintelligible, and for converting encrypted messages back into intelligible form.

DMZ: Demilitarized Zone - A part of the network that is neither part of the internal network nor directly part of the Internet. A DMZ is a network segment that resides between two other network segments.

Encryption: Encryption is the process of converting data into an unintelligible format, or “ciphertext”, so that users without the appropriate key cannot decrypt and view the information.

Public Key Cryptography: A type of cryptography in which the encryption process and one element of the encryption key is publicly available and unprotected, but in which a part of the decryption key is protected, so that only a party with knowledge of both parts of the decryption key can decrypt the cipher text to recover the original information.

Solution Life Cycle (SLC): The FSA SLC serves as a framework intended to guide a Solution Acquisition from business need to deployment and support. The SLC provides defined, repeatable and efficient processes that Solution Acquisition project managers may tailor to meet their individual needs.

SSL (Secure Sockets Layer): A protocol that supplies secure data communication through data encryption and decryption. SSL uses RSA public-key encryption for specific TCP/IP ports. The most current version of SSL is version 3.

TLS (Transport Layer Security): Transport Layer Security (TLS) is an IETF protocol, based on and nearly identical to SSL v.3, that ensures privacy between communicating applications and their users on the Internet. When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any message.

Vulnerability: Hardware, firmware, or software flaws that leave the system open for potential exploitation. Vulnerabilities usually result from a weakness in automated system security procedures, administrative controls, physical layout, internal controls, etc., that could be exploited by a threat to gain unauthorized access to information or disrupt critical processing.

3 Scope and Applicability of Standard

3.1 Scope of Web Security Standard

There is a vast body of knowledge on Web security threats, vulnerabilities, and design recommendations. This document is designed to provide a roadmap to FSA standards for Web security, and to serve as a starting point for creating or validating Web applications that provide appropriate levels of protection for FSA systems and data. This document is not intended as a comprehensive source for all information that may be needed to properly secure Web applications or Web components. References to additional resources are provided throughout the guidelines that follow each standard.

The scope of these standards include all Web technologies and components deployed in FSA systems, whether available externally (Internet Web applications) or internally (Intranet Web applications). However, implementation of specific security controls may differ depending on whether a Web application is available externally or only via an Intranet implementation.

These standards include both standalone Web applications and Web applications that have legacy system components. Although most of the guidelines address Internet and Web technologies, in some instances the configuration or deployment of other components or systems may be affected, such as middleware systems or legacy databases.

3.2 Applicability of Standard

These Web security standards apply to all FSA employees, contractors, and vendors who design, develop, or operate Web applications and Web application components on behalf of FSA.

3.3 Integration with FSA Solution Life Cycle

These Web security standards should be implemented in coordination with the FSA Solution Life Cycle (SLC). These standards are intended to supplement and not replace existing FSA risk management processes or requirements for documentation and implementation of development processes defined in the SLC. None of the recommendations or guidance provided in this document are meant to modify or negate SLC requirements.

3.4 Use of Standards and Guidelines

The design and deployment guidance provided by in this document should not be considered all-encompassing or absolute. Rather, this document should be used to define areas for consideration during design and planning for deployment of Web applications and Web components. While the following sections should prove useful to help FSA and

its contractors protect Web applications, it can not substitute for appropriate skills and experience in security design for Web systems.

These standards and guidelines are not a substitute for the FSA SLC. Both the FSA SLC and adherence to standard design practices for Web applications must also be considered for the following reasons:

- **Risk assessment requirements** – many design decisions and implementation of operational processes for specific Web applications must be based on assessment of risk inherent to the threats facing the system and the nature of the FSA services and data the system provides.
- **Changing Web threats** – new threats and forms of attack on Web applications are continually discovered. While the security guidelines define known attacks and vulnerabilities, Web application designers must also review recent sources of information about Web security to be sure they are aware of all major security issues that must be addressed.
- **Changing Web technology** – as Web technology and standards change, the security guidance provided in this document must be reviewed and updated. For example, Web Services security standards will have a significant impact on the security controls and testing protocols that will need to be incorporated into the design of Web applications. (See Section 4.4 for additional discussion of Web Services security standards.)

In short, strict adherence to the development and deployment guidelines in this section will not guarantee security of a specific Web application. Good security development principles and standard security considerations will still need to be followed during the construction and deployment of FSA Web applications. However, these guidelines will provide a starting point and references to major security concerns that should be addressed.

4 FSA Web Security Standards and Guidelines

The sections below provide guidance for the development and deployment of FSA Web applications. The major areas addressed are:

- Network and Infrastructure Security for Web Components
- Security for Web servers
- Security for Web applications
- Web Services security standards
- Encryption for Web components

This section contains standards and guidelines for development, operation, and testing of Web applications and related components. The guidance provided by this document must be interpreted in light of specific FSA business objectives, technical constraints, and operational requirements.

Each Web standard is presented by providing information in four major areas, as summarized below:

Background – The Background section provides context for the standard, such as the major threats addressed by the standard, major challenges in maintaining the security of Web applications the standard is intended to overcome, and the relationship to other standards.

Standard – The Standard section provides the high-level requirement for development, deployment, and operation of FSA Web applications. In general, the standard requires adherence to practices that will protect FSA information and information systems against common threats and modes of attack on Web applications and components of Web systems. The Standard is not meant to prescribe specific design elements for the Web systems. The standard is intended to maintain a flexible approach that will allow FSA to achieve its business goals, respond to new threats or availability of new Web technologies and designs, while providing security controls for FSA information and systems that are consistent with the overall FSA Information Technology Security and Privacy policy and external regulatory mandates.

Guidelines – The Guidelines section provides implementation details and recommendations. Typical topics covered include:

- Common Web vulnerabilities that should be addressed in the design, development, and operation of Web applications and components.
- Security issues or vulnerabilities that should be addressed in the design of Web applications and their components.
- Security issues or vulnerabilities that should be addressed by the System Security Plan for the Web application and its components.

- Security issues or vulnerabilities that should be tested during Certification and Accreditation of the Web application and its components.

Topics discussed in the Guidelines section are presented at a level of detail that provides an introduction to the security issues, but may not provide sufficient detail to allow a Web application designer or developer to mitigate a specific vulnerability. Additional details are provided in the References section, as described below.

The information provided in the Guidelines section cannot provide a comprehensive list of security threats or attacks that may conceivably be encountered when deploying or operating a Web application. Because new Web technologies are continually being introduced, and because security threats and attacks continually evolve, adhering to the guidance provided in this section is no guarantee that a Web application will achieve its intended level of protection for FSA information and systems. These guidelines provide a list of well-known attacks and common vulnerabilities, and provide a starting point for the design of security Web applications and components. They are not, however, a substitute for exercise of robust design practices that incorporate good Web application design principles, or for appropriate training and experience in security design.

References – The References section provides pointers to additional information that will be useful for Web application designers and developers. References may include Federal standards and guidelines, Internet sites for Web standards and guidelines organizations, or standard technical reference works.

4.1 Security of Network and Infrastructure for Web Components

4.1.1 Background

The network and other infrastructure components that support a Web application provide the first line of defense for preventing security breaches. By controlling network traffic and monitoring network functions for security events, infrastructure components provide critical security controls for mitigating Denial of Service (DoS) attacks and other security threats that exploit network protocols and configurations.

Design and configuration of network devices and related infrastructure components are discussed in this section. Included in the discussion are: network devices such as firewalls and routers that can enforce security policies to allow only authorized traffic; monitoring systems that can detect and respond to security attacks; and network design principles that serve security objectives.

Network security controls are important for secure deployment and operation of Web applications. Appropriate network design can mitigate many types of security threats designed to disrupt Web applications, alter application data, or gain unauthorized access to Web server and Web application components. However, network controls by themselves will not provide sufficient protection for Web applications, and they must be used in conjunction with security Web server configuration and Web application controls.

4.1.2 Standard

1. FSA Web applications will be protected by network access control devices, security monitoring systems, and other network infrastructure controls and designs appropriate to the security level of the FSA systems linked to the application and to the FSA data stored, processed, or transmitted by the application.
2. Regardless of the location where FSA Web applications are deployed, the network security designs and controls used to protect them will be reviewed and tested in a manner consistent with FSA Information Technology Security and Privacy Policy, the FSA SLC, and any other applicable federal requirements.

4.1.3 Guidelines

The network design on which FSA system components are deployed should address the major network security controls that will protect FSA systems and data. Issues that should be addressed in the network security design and security plan for Web applications include, but are not limited to, the major control areas defined below. These control areas include:

- Network Location and Design
- Network Access Control Devices and Configurations

- Network Intrusion Detection System

4.1.3.1 Network Location and Design

Networks should be segmented with firewalls and routers into appropriate security zones, with Web servers and components located to provide appropriate levels of protection.

Network location is important for Web application design for several reasons:

- Network location determines the network infrastructure components available to protect the Web server and other Web application components.
- Network location also determines what other portions of the network are vulnerable if the Web server is compromised. For example, if the Web server is located on the internal production network, then the internal network is subject to attack from the compromised Web server.
- For Web application hosting outsourced to third parties, the location of the Web server may affect the availability of administrative access to Web application components.

Several locations for Web servers and Web application components may introduce unacceptable risk. Network locations not generally recommended for Web servers and Web are:

- Internal production networks – Web servers should not be located on the same network as internal users and internal servers. This location exposes the internal network to unnecessary risk of compromise if the Web server is breached.
- In front of firewalls – placing Web servers before the firewall or router that provides IP filtering allows all network traffic to reach the Web server.

Generally, Web servers should be located in a “Demilitarized Zone” (DMZ). A DMZ is a host or network segment inserted as a neutral area between an organization’s private network and the Internet. It prevents direct access to an organization’s internal network, yet avoids the risks of either locating a Web server on an internal network or exposing it directly to the Internet. A DMZ is typically created by placing a firewall between a border router and an internal network. The firewall is configured with access control lists for network traffic that restrict the destination addresses and ports allowed for Internet traffic to only the DMZ. There are a large number of design variations on DMZ structure, each with advantages and disadvantages for security, deployment, and cost. Refer to network security design sources for detailed information on designing and configuring a network DMZ.

In general, the security advantages of a DMZ are:

- Web servers are better protected if only authorized traffic to and from them is allowed.
- Network traffic to and from the Web server can be monitored.
- Compromise of the Web server does not directly threaten the internal production network.

- DMZ network configurations can be optimized to support and protect the Web server(s).

Disadvantages of a DMZ design are:

- DoS attacks aimed at the Web server may have an effect on the internal network.
- Web server may potentially be used to attack or compromise hosts on the internal network, depending on the traffic allowed to and from the DMZ and internal network.

In most cases, a DMZ should be used to protect Web servers. See Section 4.2.2 for additional design considerations for protecting communications between multiple Web servers or other components of a Web application.

4.1.3.2 Network Access Control Devices and Configurations

This section addresses security design for network devices issues in two major areas:

- Restricting network traffic to authorized protocols, services, and addresses
- Monitoring network traffic to detect potential attacks

Firewalls and Routers

Network access controls are typically implemented with routers and firewalls that can filter traffic based on Access Control Lists or other criteria. A wide range of traffic filtering capabilities have been developed. The simplest approach is to use routers or firewalls to allow or deny network packets based on protocol, source or destination addresses, or source or destination IP ports. More sophisticated techniques can monitor “sessions” by tracking related requests and responses (“stateful inspection” firewalls) and application proxies that inspect application-level packet contents. Newer firewall products may combine different approaches, and may even include intrusion detection functions that produce alerts when suspected attacks are detected, or “intrusion prevention” systems that can take active measures to block traffic or otherwise respond when attacks take place.

Packet inspection firewalls receive incoming requests and attempt to match the header portions of packets (along with other criteria) with firewalls rules of access policies. If the traffic meets the criteria for an ‘allowed’ rule, the packets are passed through the firewall to the requested destination. If the traffic matches a ‘deny’ rule, or they don’t match any of the ‘allowed’ rules, the packets are rejected or dropped. Packet inspection firewalls can be divided into two additional categories: stateful and non-stateful. A stateful packet inspection firewall tracks session characteristics when the session is initiated, and can make decisions about allowed types of response without requiring explicit rules for return traffic. Without this capability, the outbound and inbound rules must both be defined.

Proxy firewalls receive a network packet on one interface, inspect the packet in the application layer, reconstruct the packet, then forward the packets out through another

interface. Proxy-based solutions can easily enforce user authentication requirements to force users to log in to the proxy before a request is serviced. This provides more effective control than just basing access decision on the source TCP/IP address.

Regardless of which firewall technology is chosen (including a combination of different types), there are general rules that should be applied:

- Restrict traffic between Web clients and Web content servers by allowing only external inbound connections to be formed over ports 80 and 443. Additional firewall rulesets may be required to pass traffic between Application Servers and RDBMS engines such as port 1521.
- Implement a ‘default deny’ policy, as discussed in the next section.

Commonly Attacked Ports to Consider Blocking³

When considering access control policies for firewalls and routers, the recommended approach is to block ports that don’t have specific requirements to be open. This is commonly referred to as a ‘default deny’ policy, whereby the default rule is to deny all traffic to all ports, then to allow traffic for specific protocols as needed for business purposes. However, some TCP/IP ports are much more commonly attacked than others. The table in Appendix A lists ports that are commonly probed or attacked. While this list does not constitute a comprehensive firewall rule base, it does provide guidance for blocking specific ports that represent known vulnerabilities. Blocked ports should still be monitored to detect attacks or potential intrusion attempts. Note that blocking some of these ports may have potential effects on existing systems and application communications, so any rule changes that block ports must be tested before making changes.

4.1.3.3 Network Intrusion Detection Systems

An Intrusion Detection System (IDS) is software that monitors system and network traffic to determine when a system may be under attack. IDS software observes network traffic, system resources, and other events, along with knowledge of attack characteristics, then notifies network administrators and appropriate security personnel when a possible intrusion or penetration attempt is identified.

The two principal types of IDSs are host-based and network-based. Network-based IDS uses a network traffic sensor to monitor all transmissions on a network segment, searching for signs of attack or penetration attempts. Most network IDSs rely on predefined “attack signatures” to detect and identify attacks. Attack signatures are a series of events or traffic patterns usually present when specific types of attack or penetration attempts are in progress. Network-based IDSs can monitor multiple hosts or multiple network segments simultaneously. Network-based IDS are typically installed on a dedicated host, so they do not affect performance of Web servers. Another advantage is that they are not compromised by a successful attack on a Web server itself.

³ Based on the SANS Twenty Most Critical Internet Security Vulnerabilities (2003)

Network-based IDSs have limitations because of their design. Attackers can format the details of an attack (e.g., fragment packets, alter attack pattern so that it does not match the attack signature, spread the attack out over time) so that it is not recognized by the network-based IDS. Some network configurations, such as the use of switches, can affect the ability of a network-based IDS to detect attacks. Network-based IDS are also more susceptible to being disabled by DoS attack (even those not directly targeted at the IDS).

Host-based IDSs must be installed on each individual computer system that is to be monitored or protected. Host-based IDSs must be integrated with the operating system they protect, so it must be designed specifically for each operating system. These types of IDSs monitor network traffic to and from the host, the use of system resources, and system log files. Host-based IDSs are particularly useful when network traffic to and from the Web server is encrypted (e.g., when SSL/TLS is in use).

Because Host-based IDSs are located on the server, they can detect some attacks and penetration attempts not recognized by network-based IDSs. However, host-based IDS may have a negative effect on host performance. In general, the greater the detection capabilities, the greater the negative impact on the performance of the host. Host-based IDSs may not detect some network-based attacks such as certain DoS attacks.

Both host-based IDSs and network-based IDSs share some weaknesses. The most significant weakness is no IDS can detect all, or, often, most, of the attacks that exist today. In addition, IDSs require frequent updates to their attack signature databases in order to recognize new attacks. An IDS that is not updated frequently will fail to recognize the newest forms of attack. Another weakness of IDS is that they often produce false positives, sounding alarms when no attack is actually in progress. IDS rules can be tuned based on experience to decrease the number of false alarms, but this approach typically requires significant effort by trained resources to optimize the IDS rule base and to monitor and analyze IDS alerts to determine if they represent real security events.

4.1.4 References

1. NIST Special Publication 800-44, *Guidelines on Securing Public Web Servers*, September 2002
2. NIST Special Publication on Intrusion Detection System, *Intrusion Detection Systems*
3. NIST Special Publication 800-64, *Security Considerations in the Information System Development Life Cycle*, October 2003
4. Security Administration, Networking, and Security (SANS) Institute, *The Twenty Most Critical Internet Security Vulnerabilities*, <https://www.sans.org/top20>, October 2003

4.2 Security of Web Servers

4.2.1 Background

Web servers have proven a frequent site of attack to compromise Web applications. Rapid version changes, the introduction of numerous new features and scripting capabilities, and the complexity of Web server software have all contributed to the proliferation of vulnerabilities that can be exploited by Internet attackers.

Attacks on a Web server may have several objectives:

- Change the content of information displayed to users.
- Gain access to sensitive data and business functions.
- Exploit the Web server to gain access to other internal systems and data.
- Mislead Web users to gain information or falsify transactions for financial gain.

Many of the security vulnerabilities associated with Web servers result from configuration issues. For example, Web server default installations may enable unneeded services, install example programs and scripts that can be exploited, or fail to apply access controls to key files and directories. Recognizing the importance of security Web server configuration, vendors have started introducing more security default configurations and offering tools to aid in security testing and deployment. Many of the security vulnerabilities discussed in this section are specific to individual products or vendors, and may require use of specialized tools or configuration scripts. The References section provides additional resources to address configuration requirements and tools that are available for testing and managing proprietary Web server software.

4.2.2 Standard

1. Web server operating systems and other software components deployed to support FSA Web applications will be configured through use of security hardening practices to minimize vulnerabilities to Web security threats in a manner consistent with the security level of the FSA systems linked to the application and the FSA data stored, processed, or transmitted by the application.
2. The design and configuration of Web server operating systems and supporting software components will be reviewed and tested in a manner consistent with FSA Information Technology Security and Privacy Policy, the FSA SLC, and any other applicable federal requirements.

4.2.3 Guidelines

The Web Server security guidelines included here are:

- Securing Operating Systems for Web Servers
- Operating System Security

- Server-Server Trust in Web System Design

4.2.3.1 *Securing Operating Systems for Web Servers*

Most commonly available Web servers run on a general-purpose operating system. Many security issues can be avoided if the operating systems that control Web servers are configured to remove vulnerabilities or mitigate risks.

Default hardware and software configurations are typically set by vendors to emphasize features, functions, and ease of use at the expense of security. Because vendors are not aware of each organization's security needs, each Web administrator must configure new servers to reflect their organization's security requirements and reconfigure them as those requirements change. The practices recommended here are designed to help Web administrators configure and deploy Web servers that satisfy their organization's security requirements. Web administrators with existing Web servers should confirm that their current configurations address the issues discussed here.

Techniques for hardening different operating systems vary greatly, so this guidance will only present generic procedures common in securing most operating systems. Resources for securing specific operating systems are provided in the References section. There are also automated tools for hardening operating systems, and we recommend considering the use of such tools and others with similar functionality.

Four basic steps are necessary to maintain basic operating system security:

- Planning, installing, and deploying the Web server operating system
- Configuring the Web server operating system to adequately address security
- Patching and updating the Web server operating system as required
- Testing the Web server operating system to ensure that the previous three steps are adequately addressing all security issues

These topics are discussed in greater detail below.

4.2.3.2 *Operating System Security*

Patching and Upgrading Operating Systems

Typically, operating systems as supplied by the vendor have known vulnerabilities that need to be corrected before using the operating system to host a Web server. Once an operating system is installed, vendor-provided fixes should be applied to correct for these known vulnerabilities. The permanent fixes may be called patches, hotfixes, service packs, or updates.

To adequately detect and correct for these vulnerabilities, Web developers or administrators should:

- Create and implement a patching process
- Identify vulnerabilities and applicable patches
- Mitigate vulnerabilities (until patches are available, tested, and installed)

To check for operating system or Web server application vulnerabilities, see the NIST ICAT Metabase at <http://icat.nist.gov>. Guidelines on Securing Public Web Servers.

Remove or Disable Unnecessary Services and Applications

Ideally, a Web server should be on a dedicated, single-purpose host. Many operating systems are configured by default to provide a wider range of services and applications than required by a Web server. Web developers and administrators should configure the operating system to remove or disable services not needed to support the Web server.

Some common examples of services that should usually be disabled would include:

- Windows Network Basic Input/Output System (NetBIOS)
- NFS, if not required
- File Transfer Protocol (FTP)
- Berkeley “r” services (e.g., rlogin, rsh, rcp)
- Telnet
- Network Information System (NIS)
- Simple Mail Transfer Protocol (SMTP)
- Compilers
- Software development tools

If possible, remove unnecessary services and applications instead of simply disabling them through configuration settings. Attacks that attempt to alter settings and activate a disabled service cannot succeed when the software components required for them to run are absent.

Eliminating or disabling unnecessary services enhances the security of a Web server in several ways⁴:

- Unnecessary services cannot be compromised and used to attack the host or impair the Web server services. Each service added to a host increases the risk of compromise for that host because each service is another possible avenue of access for an attacker. Less is truly more in this case.
- Different individuals may administer different services. Isolating services so each host has a single administrator will minimize the possibility of conflicts between the administrators. Also, having a single administrator responsible for a host provides better accountability. For more information on vulnerabilities and patching, see NIST Special Publication 800-40, Procedures for
- Handling Security Patches (<http://csrc.nist.gov/publications/nistpubs/index.html>).
- The host can be configured to better suit the requirements of the particular service. Different services might require different hardware and software

⁴ Security Network Servers, Computer Emergency Response Team (CERT), 2000, <http://www.cert.org/security-improvement/modules/m10.html>

- configurations, which could lead to unnecessary vulnerabilities or service restrictions.
- By reducing services, the number of logs and log entries is reduced; therefore detecting unexpected behavior becomes easier.

When configuring the operating system, apply the general security principle of “disable everything except that which is expressly permitted.” Disable or, preferably, remove all services and applications, then selectively enable only those required by the Web server. Install the minimal operating system configuration that is required for the Web server application to function normally. Note that many uninstall scripts or programs do not completely remove all components of a service; so, it is usually better to avoid installing unnecessary services in the first place.

The services enabled on a Web server will depend on the functions the organization wants the server to provide. Those services may include database protocols, file transfer protocols, and remote administration services. Each of these functions, even though they may be required, increases the security risk to the server. Whether the risks outweigh the benefits is a decision that must be based on the business objective for the Web system and the data processed or stored by the system.

Configuring Operating System User Authentication

To enforce policy restrictions on who can access Web servers and the server operating system, user authentication must be configured to control access. Even if Web server is publicly available, administrative and specialized access to Web servers and other Web components should be limited to specifically authorized individuals and groups that are responsible for configuration.

Setting up user authentication usually involves configuring parts of the operating system, firmware, and applications on the server, such as the software that implements a network service. These configurations should follow the FSA Security and Privacy Technical Architecture⁵. This architecture is also detailed in Appendix D. In special cases, for high-value or high-risk sites, it may be necessary to also use authentication hardware, such as tokens or one-time password devices. Use of these devices should adhere to Federal Information Processing Standards (FIPS), such as FIPS 140-1 and FIPS 140-2.

The following steps are recommended to protect the Web server from unauthorized modification or configuration:

- Remove or disable unneeded default accounts and groups. This includes default guest accounts, administrator or root level accounts, and accounts associated with local and network services. This step eliminates potential misuse of these accounts by intruders.
- Disable noninteractive accounts. Disable accounts (and the associated passwords) that need to exist but do not require an interactive login. For Unix systems,

⁵ Security and Privacy Architecture Specification, Deliverable 124.1.3, May 2003.

disable the login shell, or provide a login shell with NULL functionality (/bin/false).

- Create user groups and assign users to the appropriate groups. Define rights to the groups, then administer user rights by assigning or removing membership in the appropriate groups instead of by assigning rights to individual users.
- Create only the necessary user accounts. Discourage or prohibit the use of shared accounts.
- Configure password setting in accordance with the FSA password policy defined in the FSA IT Security and Privacy Policy. Set account passwords appropriately. The password policy should address the following issues: password length, password complexity, password aging, password reuse, and resetting passwords.
- Configure computers to deny login after a small number of failed attempts. However, failed network login attempts should not prevent an authorized user or administrator from logging in at the console. All failed log in attempts whether via the network or console should be logged. If remote administration is not required, disable the ability for the administrator or root level accounts to log in from the network.
- Install and configure other security mechanisms to strengthen authentication. Consider using other authentication mechanisms such as tokens, client/server certificates, or one-time password systems.
- Generate and distribute user account reports. These reports should be disseminated to appropriate supervisors and management personnel to identify individuals who no longer require accounts.

4.2.3.3 *Configure Web Server Resource Controls*

Most operating systems provide the ability to specify access privileges individually for files, directories, devices, and other system resources. Careful design and configuration of access controls can reduce intentional and unintentional security breaches. For example, denying read access to files and directories helps protect confidentiality of information, and denying unnecessary write or modify access can protect the integrity of information. Limiting the execution privilege of most system-related tools and utilities to only authorized administrators helps prevent users from making configuration changes that could reduce security levels. It can also restrict the ability of intruders to use those tools to attack the system or other systems on the network. Operating system resource controls act in tandem with Web server resource controls.

4.2.3.4 *Security Testing the Operating System*

Periodic security testing of the operating system is important to identify vulnerabilities and to ensure that the existing security precautions are effective. Typically, security testing of the operating system will be part of the overall Certification and Accreditation process for a Web application.

Of the several methods for testing operating systems, the most common are vulnerability scanning and penetration testing. Vulnerability scanning usually entails using an automated vulnerability scanner to scan a host or groups of hosts on a network for application, network, and operating system vulnerabilities. Penetration testing is a testing

process designed to compromise a network using the tools and methodologies of an “attacker.” Both of these testing techniques are also applicable to testing Web server applications.

4.2.4 Server-Server Trust in Web System Design

Security of communications links between components of a Web application must be addressed to define end-to-end security controls. The typical approach is to define (either explicitly or implicitly) the trust relationships between servers: Web server to application server, Web or application server to legacy system, application server to database, etc. Trust relationships primarily define the authentication mechanism that allows each component in a Web application to assure itself that an incoming request is from an authorized server, and that responses are sent to authorized servers. Confidentiality of the communications link may also be considered in the trust relationship, and encryption of traffic may be used to protect both the confidentiality and integrity of information transferred.

The table below outlines common security controls for authenticating communications links and protecting the confidentiality and integrity of data transmitted. A combination of controls can be used to provide multiple modes of protection. Design of end-to-end security controls for Web applications must consider the trade-offs between the sensitivity of the data, risks associated with the data, and costs of system deployment and operation, including effects on performance.

Security Mechanism	Control Objectives	Description and Benefits	Impact and Disadvantages
Password Sever-to-Server Authentication	Establish trust, authorize transaction	<ul style="list-style-type: none"> Requires that a password be provided by the requesting server when opening a session or sending a transaction request Relatively simple to implement Relatively low cost authentication mechanism 	<ul style="list-style-type: none"> Passwords may be intercepted if transmitted in clear text Stored passwords must be protected Existing applications may need to be modified to send or validate passwords Passwords authentication subject to many well-known weaknesses
Digital Certificate Sever-to-Server Authentication	Establish trust, authorize transaction	<ul style="list-style-type: none"> Provides strong authentication between servers Can be implemented with the SSL/TLS protocol using client-side certificates If SSL used, also provides encryption of communications link 	<ul style="list-style-type: none"> Digital certificates must be created or purchased and managed Increases complexity of system testing and maintenance SSL/TLS between components may affect overall system

Security Mechanism	Control Objectives	Description and Benefits	Impact and Disadvantages
Kerberos Server-to-Server Authentication	Establish trust, authorize transaction	<ul style="list-style-type: none"> Provides secure, encryption-based authentication using the well-established Kerberos protocol Kerberos available natively in Microsoft Windows server operating systems 	<p>performance</p> <ul style="list-style-type: none"> If not available natively, additional software and hardware costs may be incurred to deploy a Kerberos authentication server Use of Kerberos authentication between system components may affect performance of overall system Kerberos does not natively provide encryption for transmitted data
Network Access Control	Establish trust relationship	<ul style="list-style-type: none"> Network segmentation with routers and firewalls can establish network zones to isolate Web application components Router or firewall access policies can restrict traffic between specific servers based on IP address and port 	<ul style="list-style-type: none"> Authentication of servers based on IP address is not as strong as other mechanisms Traffic filtering may decrease system performance Router or firewall filtering may interfere with other network functions such as load balancing or address translation
Encryption	Protect confidentiality of communications link	<ul style="list-style-type: none"> Encryption of data transmitted between system components prevents interception of sensitive data or transaction details Encryption may be implemented with either SSL/TLS protocols or using special-purpose encryption modules 	<ul style="list-style-type: none"> Encryption keys or digital certificates used for encryption will need to be managed If SSL/TLS will not be used, add-on encryption software may incur additional costs Encryption by itself does not provide authentication of system components Encryption between components may affect overall system performance

Figure 1- Security Mechanisms to Protect Server-to-Server Communications

4.2.5 References

1. NIST Special Publication 800-44, *Guidelines on Securing Public Web Servers*, September 2002
2. NIST ICAT Metabase at <http://icat.nist.gov> provides Guidelines on Securing Public Web Servers
3. Security Network Servers, Computer Emergency Response Team (CERT), 2000, <http://www.cert.org/security-improvement/modules/m10.html>

4.3 Security of Web Applications

4.3.1 Background

Many of the most effective and difficult to prevent attacks on Web applications do not rely on direct network or Web server attacks. Rather, they use standard http requests that do not trigger access control rules, but manipulate or subvert normal data input functions that compromise the Web application itself. Such attacks may insert unauthorized commands into data input forms, take advantage of buffer overflows or other common Web application vulnerabilities to execute arbitrary commands, or exploit Web application functions by altering parameters passed to internal system components. These attacks are not usually mitigated by simple traffic filtering or by searching for attack signatures. Addressing Web application vulnerabilities requires knowledge of Web application design principles and use of good coding practices for Web components. This section discusses the major vulnerabilities that must be considered when developing Web applications.

4.3.2 Standard

1. FSA Web applications systems and other software components deployed to support FSA Web applications will be designed, configured, and deployed to minimize vulnerabilities to Web security threats in a manner consistent with the security level of the FSA systems linked to the application and the FSA data stored, processed, or transmitted by the application.
2. The design and configuration of FSA Web applications and supporting software components will be reviewed and tested in a manner consistent with FSA Information Technology Security and Privacy Policy, the FSA SLC, and any other applicable federal requirements.

4.3.3 Guidelines

While there are numerous types of security vulnerabilities that can potentially serve as avenues for attacking Web applications, there is a relatively small set of common Web application design issues that have proven most significant. These common vulnerabilities are described in this section. Other major areas of Web application security design are also described.

These guidelines are not meant as an exhaustive list of issues that should be addressed, or of all possible security vulnerabilities. Web application designers cannot assume that addressing these vulnerabilities will prevent all possible security problems. Web application designers and those responsible for operating and maintaining applications will need to continually monitor the latest forms of attack as they develop. However, because the vulnerabilities described in this section are well known, they represent the minimum set of security design issues and vulnerabilities that must be addressed during development and deployment. Specifically, these issues should be addressed during

initial design of a Web application and during development of a System Security Plan. Many, although not all, of the issues discussed in this section may be tested using standard Web application security testing tools and procedures.

The major areas of concern for security design in Web applications addressed in this section are:

- Web authentication and access control
- Account configuration and session management
- Input and parameter validation
- Cross-site scripting
- Command and SQL injection
- Buffer overflows
- Error handling
- Remote Procedure Calls (RPC)
- BIND Domain Name System
- Microsoft SQL Server (MSSQL)
- General HTML and design considerations
- Encryption
- Remote administration
- Middleware security
- Database security

The sections below briefly describe these vulnerabilities and common approaches for addressing them. The additional Web application security sources in the reference section should be consulted for detailed advice on designing, coding, and testing programmatic controls and configurations to mitigate the risks associated with these vulnerabilities.

4.3.3.1 Web Authentication and Access Control

Access control mechanisms are a necessary and crucial design element to any application's security. In general, a Web application should protect front-end and back-end data and system resources by implementing access control restrictions on what users can do, which resources they have access to, and what functions they are allowed to perform on the data. Ideally, an access control scheme should protect against the unauthorized viewing, modification, or copying of data. Additionally, access control mechanisms can also help limit malicious code execution, or unauthorized actions through an attacker exploiting infrastructure dependencies (DNS server, ACE server, etc.).

To the extent possible, Web applications should take advantage of the FSA Security and Privacy Architecture for guidance in designing and deploying authentication and authorization controls. Architecture components may provide common definitions, design solutions, or security services that can be called by Web applications. For example, an FSA Single Sign-on service is under development. Use of standard FSA security services will:

- Simplify Web application design and development.
- Provide consistent controls for common security functions
- Decrease time and cost associated with development of security functions.

4.3.3.2 Account Configuration and Session Management

Because HTTP is a stateless protocol, maintaining state during a browsing session is a common requirement. Typically, browsers store and send to the Web server a session identifier previously received from the server. The session id is stored in the URL, in a cookie or, occasionally, in a form field. The session id must be a non-guessable value in order to prevent session hijacking and we will discuss the issue in the section on randomness. Unless appropriate controls are in place, session data should not be stored in the browser with the expectation that it will be passed untouched back to the server.

Session information should be protected to avoid session hijacking or modification of session information. Example controls include:

- Session tokens (identifier strings or cookies) should be encrypted or hashed to prevent forgery.
- Session tokens should not be predictable.
- Session tokens should be periodically expired to force frequent renewal to avoid use after interception.

4.3.3.3 Input and Parameter Validation

Most of the common attacks on systems (whose descriptions follow this section) can be prevented, or the threat of their occurring can be significantly reduced, by appropriate data validation. All data or parameters passed to Web applications should be validated for to establish bounds on length, content, and use of special characters or command strings. The goal is to prevent parameter tampering or entry of command strings that could be used to compromise Web applications.

Effective data and parameter validation requires careful formulation of rules and rule-checking logic. As an alternative to building these capabilities into Web applications, commercial add-on systems are available that provide a data validation “shell” around Web applications. These systems provide data filtering and validation functions that can be deployed as infrastructure services for Web applications without requiring changes in the Web application itself.

4.3.3.4 Cross-site Scripting

Cross-site scripting (XSS) is an attack that uses a Web application vulnerability to send harmful code (generally JavaScript) to an end user. Web applications that utilize unfiltered user input to produce an output are susceptible to this attack. An attacker can insert malicious code in the input and the application transmits the output to other users. This attack exploits the trust placed by the end user in the Web application.

Cross-site scripting attacks can be categorized into *stored attacks* and *reflected attacks*. In stored attack the malicious code is permanently stored in some form on the Web server or other associated systems. In a reflected attack the malicious injected code takes a different route to the user, such as e-mail. When a user is tricked into sending this malicious code, it travels to the affected Web server and reflects back to the user's browser. Since the user trusts the server, this code is executed by user's browser.

The best method of protecting a Web application from XSS attacks is to have a detailed code review that searches the code for validation of all headers, cookies, query strings, form fields, and hidden files against a rigorous specification of what should be allowed. A complete description of how to avoid XSS attacks is beyond the scope of this standard. Consult the references for additional guidance.

4.3.3.5 Command and SQL Injection

Command and SQL injection attacks can relay harmful code from the Web application to other systems. Scripts written in *perl*, *python* and many other languages can be injected and executed by poorly designed Web applications. A Web application is susceptible to this type of attack when it uses an interpreter of any type.

SQL injection attacks utilize a parameter that the Web application passes on to a database. An attacker can modify the SQL parameter and trick the Web application into forwarding a harmful query to the database.

The best approach to protect against this vulnerability is to avoid using command injection. The risk associated with command injection can be removed by avoiding the use of operating system shell interpreters. For the backend databases calls that still need to be employed, careful validation of the data used to construct a query to be sure it does not contain malicious content.

4.3.3.6 Buffer Overflows

A carefully selected input to a Web application can corrupt the execution stack of the application. This causes the application to execute arbitrary code, allowing an attacker to effectively take control of the machine. Buffer overflow flaws can be present in both Web server or application server products or in the Web application itself. Buffer overflows can also be found in custom Web application code. Almost all the known Web servers, application servers, and Web application environments are susceptible to buffer overflow attacks. The notable exceptions to this are Java and J2EE environments.

The best way to protect against buffer overflow attacks is to monitor the latest bug reports on the products used in the Internet infrastructure. The application of recent patches, combined with data validation techniques described above, will reduce the buffer overflow threat.

4.3.3.7 Error Handling

The improper handling of errors can give rise to a variety of security problems for Web applications. The most common error handling mistake involves the inappropriate display of detailed internal error messages to an attacker. These messages may include, but are not limited to, stack traces, database dumps, and error codes. These messages may reveal physical paths of files and internal architecture details of the website that can be used by an attacker to gain unauthorized access.

An error handling scheme needs to be developed to gracefully handle errors generated by Web applications without revealing information useful to attackers. The scheme should provide meaningful error information to the users and maintenance information to website administrators, but should not provide helpful information for attackers. User error messages should be logged to allow for review by administrators. Review of error message may disclose vulnerabilities in the Web applications error handling mechanism.

4.3.3.8 Remote Procedure Calls (RPC)

RPCs allow programs on one computer system to execute procedures on another computer by passing data and retrieving the results. RPC is widely used for many distributed network services such as remote administration, NSF file sharing, and NIS. But many RPC services execute with elevated privileges and provide an attacker unauthorized remote root access to the vulnerable system. Attackers have exploited RPC vulnerabilities to launch distributed Denial of Service (DoS) attacks on many major Web sites.

The best approach to protect against this attack is to shut down or remove RPC services that are not essential for the proper functioning of the network. The application of the latest vendor patches to Web application components can also reduce the threat of attacks based on RPC services.

4.3.3.9 BIND Domain Name System

Berkeley Internet Name Domain (BIND) package is the most widely used implementation of the Domain Name Service (DNS). DNS systems facilitate the conversion of hostnames (e.g.; www.ed.gov) into registered IP addresses. The ubiquity and critical nature of BIND has made it a frequent target, especially in Denial of Service attacks, which can result in a complete loss of accessibility to the Internet for services and hosts. Most of the BIND vulnerabilities are a result of misconfigured DNS systems and lack of awareness by administrators about security updates.

Protection against BIND vulnerabilities can be achieved by disabling the BIND daemon *named* on any system which is not specifically designated and authorized to be a DNS server. System administrators should apply the latest vendor patches to the DNS servers to mitigate the risk of BIND flaw attacks.

4.3.3.10 Microsoft SQL Server (MSSQL)

The Microsoft SQL Server (MSSQL) contains several serious vulnerabilities that allow remote attackers to obtain sensitive information, alter database content, compromise SQL servers, and, in some configurations, compromise server hosts. MSSQL vulnerabilities are well-publicized and actively under attack. Two recent MSSQL worms in May 2002 and January 2003 exploited several known MSSQL flaws. Hosts compromised by these worms generate a damaging level of network traffic when they scan for other vulnerable hosts.

Steps recommended for protection against these attacks include:

- Disable SQL/MSDE Monitor Service on UDP Port 1434.
- Apply the latest service pack for Microsoft SQL/MSDE server and/or MSDE 2000.
- Apply the latest cumulative patch that is released after the latest service pack.
- Apply any individual patches that are released after the latest cumulative patch.
- Enable SQL Server Authentication Logging.
- Secure the server at system and network level.
- Minimize privileges of the MSSQL/MSDEServer

4.3.3.11 Remote Administration

Administrative interfaces provide powerful Web application managing capabilities. These features allow administrators to manage users, data and the site content. Complex Web sites support granular administration. These administrative interfaces are prime targets for both internal and external attackers.

A lack of strong authentication to these interfaces might lead to unauthorized access. Implementation of weak encryption methods might allow attackers to compromise these interfaces. The flaws in the separation mechanism between users and administration can compromise the security of Web applications. Providing unnecessary functionality for all administrators can also risk the security of Web applications.

The primary method for protecting against remote administration flaws is to never allow administrator access through the front door if possible. Given the power of these interfaces, most organizations should not accept the risk of making these interfaces available.

4.3.3.12 Middleware Security

Middleware provides communications between elements of Web applications. Middleware components are often employed to provide links between Web applications and mainframe or legacy applications and databases. Security for middleware addresses the controls for securing data transmitted between Web application servers, other system components, and application users. A good middleware security management solution should provide functionality for authentication, authorization and audit. Some of the middleware technologies that can be used with Web applications include:

- COM/COM+/.NET
- Enterprise Java Beans (EJB)
- CORBA
- DCOM
- Remote Method Invocation (RMI)
- Internet Inter-Orb Protocol (IIOP)
- CSIv2
- SOAP

A complete description of all security issues related to middleware used for Web applications is beyond the scope of this standard. Refer to vendor guidance for methods to secure these components.

4.3.3.13 Database Security

Database security involves these major security functions:

- Authentication
- Authorization
- Confidentiality
- Integrity
- Accountability

Most of the internal and external attacks on Web application databases exploit flaws in one of the services listed above. The lack of user input validation can provide security risks for the database layer. Since not all data should be need be treated in the same manner, granular access controls to restrict access to the data reduces the risk of database compromise. When possible, database administrators should implement and enforce read only rights to tables and forms within the database. Such restrictions can provide additional security controls if the previous Web application layers have failed to prevent an attack.

4.3.3.14 General HTML and Security Design Considerations

Use POST, Not GET, for User Input

When sensitive data is to be passed to the server, do not send it as a parameter in the query string using the GET command. This is not appropriate because parameter that are passed in the GET request are logged in plain text by the Web server, as well as in whatever proxies might be on the way. Also, the entire URL may be stored by the browser in its history, potentially exposing the sensitive information to a later user of the same machine.

In contrast, the POST method uses the HTTP body to pass information. This is preferable because the HTTP body is not normally logged. However, by itself the POST method does not offer sufficient protection. Data confidentiality and integrity may still be at risk because the information is still sent in clear text (or quasi clear text as in Base64 encoding). This means that encryption should still be used for sensitive information.

Do Not Store Sensitive Information in JSP or ASP Pages

Java Server Pages (JSP) and Active Server Pages (ASP) allow programmatic display of dynamic html content in Java and Microsoft environments, respectively. The practice of hardcoding credentials (such as username/password) leads to several types of potential security issues. Some security problems may cause a Web server to display the source code of an JSP or ASP page to be displayed instead of being executed. This means that credential information could be revealed. Credential information should be stored in a centralized location and called when needed. This allows audit logging to track access to credentials.

Remove Sensitive Data from HTML and Script Comments or Source Code

Comments that contain sensitive system or security data should not be embedded in HTML or client scripts. These comments can reveal useful information to an attacker. An example of this would be a connection string that was once part of the server side script, then commented out. In time, through inadvertent editing, this information could appear in the client script and thus be transmitted to the browser.

4.3.4 References

1. The Open Web Application Security Project (OWASP), *The Ten Most Critical Web Application Security Vulnerabilities*, January 2003
2. The Open Web Application Security Project (OWASP), *A Guide to Building Secure Web Applications*, September 2002
3. Security Administration, Networking, and Security (SANS) Institute, *The Twenty Most Critical Internet Security Vulnerabilities*, <https://www.sans.org/top20>, October 2003
4. Vendor sites, guidelines, and tools: all major vendors of Web server software provide guidelines and/or tools to assist with secure configuration of their products. Some prominent examples:
 - Apache Web servers – http://httpd.apache.org/docs/misc/security_tips.html
 - IBM Web servers – <http://www-106.ibm.com/developerworks/security/library/s-wssec.html>
 - Microsoft Web servers and components – <http://msdn.microsoft.com/security/>

4.4 Web Services Security Standards

4.4.1 Background

Web services provide standards and protocols for sharing application functions both internally and externally across the Internet. This relatively new approach to distributed computing offers the promise of more modular business applications that promote sharing of information and more rapid deployment of business services. But security has been identified as a major hurdle in the implementation of major Web services. Some of the security challenges for deployment of Web services include:

- **Secure communication channels** – FSA must be able to demonstrate that communications between business applications are not intercepted.
- **Verification of processing entity** – FSA must establish trust that a Trading Partner performing some business logic is not an imposter.
- **Verification of requesting entity** – FSA must establish trust that a Trading Partner requesting a Web service is not an imposter.
- **Management of security privileges** – FSA must effectively manage security rights of systems and users that are not in the same location.
- **Server to server security** – FSA must establish automated trust between servers that provide Web services.(see Section 4.2.4)

Web services security standards are under development. When adopted, they will provide many business and technical benefits, as listed in the following tables.

Benefit	Description
Reduced costs of supporting multiple security architectures	Supports reuse of security by multiple platforms and architectures. Reduces security costs associated with integrating with partners or rolling out new applications.
Increased secure communication channels with partners	Leveraging standardized security mechanisms results in increased interoperability with business partners systems without expensive integration costs.
Distributed Security Cost Reductions	Security services that are costly to implement and maintain can be outsourced / delegated to service providers or trusted partners.

Figure 2 - Business Benefits of Web Services Security

Benefit	Description
Secure Communication	Ability to ensure that Web Services communications are not being intercepted and compromised.
Authorization and Authentication	Ability to evaluate entities involved in Web service transactions and determine their identities and rights / Single sign-on.
Administration	Web Services Security technologies offer the benefit of allowed delegated and decentralized security management.
PKI with less overhead	XML Key Management Standard (XKMS) provides enhanced security with reduced deployment/ management overhead.

Figure 3 – Technical Benefits of Web services security

There is also a federal effort to implement Web services components as part of imitative to create a federal security architecture. One of the most prominent efforts is the eGov initiative which has been developing an eAuthentication capability. The General Services Administration (GSA) is managing the e-Authentication initiative. The goal of the GSA is to establish a gateway to provide common government wide authentication services.

The e-Authentication initiative is continuing work on developing authentication architecture for the federal government. Authentication architecture guidelines will be published by GSA in December 2003. The architecture description is expected to include standards for sharing credentials and defining authentication levels. The e-Authentication initiative will also propose a method to identify authentication levels for government agency applications.

4.4.2 Standard

1. FSA will monitor development of Web services security standards in coordination with Web services development projects.
2. When Web services security standards gain broader acceptance, FSA will adopt specific standards and guidelines for Web services security that are consistent with the FSA Security and Privacy Architecture and the FSA Enterprise Architecture.
3. FSA should secure existing Web services with currently available technologies. For example, leverage SSL to secure point to point communications for FSA Web services.

4. FSA will identify additional security controls and evaluate the applicability of available Web services security standards as new Web service functionality is deployed.
5. FSA should identify security integration points with Trading Partners. FSA should identify how its Trading Partners are securing the Web services provided by them. These security integration points will need to be defined in conjunction with the Trading Partners.

4.4.3 Guidelines

This section summarizes the current status of the following Web services security standards:

- SSL/HTTPS
- XML-Encryption
- XML-Signature
- XKMS
- SAML
- XACML
- WS-Security
- WS-Security Extensions

To follow the development of Web services security standards, the activities of major standards development organizations should be monitored. Oasis and WS-I are guiding the major effort to define standards around Web Services security technologies for SAML and WS-Federation standards, respectively. The Liberty Alliance is developing protocols that address a wide range of technical, policy, and governance standards to support implementation of federated identity standards for business services. FSA should follow the progress of these groups to better understand when Web services security standards are mature enough to implement.

Evaluate the security SDKs available for software. Major vendors of security products have already released SDKs that support new Web services security standards (WS-Security, SAML, XML-sig, XML-enc). For example, web access control software that provides extranet access management services supports Web services standards in their current versions or will in their next releases.

Figure 4 below provides an overview of Web services security standards that are in development. The diagram also summarizes expected usage for each standard.

Technology / Standard	Security Benefit	Description	Expected Usage
SSL / HTTPS	Confidentiality Integrity Authentication Non-repudiation	SSL is an existing TCP/IP security protocol used to secure Web communication at the transport layer.	SSL can be used to encapsulate and protect Web Services communications from point to point.
XML-Encryption	Confidentiality Integrity	Standard for encrypting the payload of XML SOAP messages.	Parts of an XML document can be encrypted.
XML-Signature	Integrity Authentication Non-Repudiation	Standard for generating a hash and signing XML SOAP messages.	Parts of an XML document can be digitally signed.
XKMS	Management	XML Key Management Standard – a specification that enables Web services to register and manage cryptographic keys used for digital signatures and encryption.	Thin clients (can obtain key information (values, certs) to enable secure end-to- end communications.
SAML	Authentication Authorization	SAML (Security Assertion Markup Language) is a standard for that enables the exchange of authentication and authorization information.	SAML defines assertions that authorize an entity to perform actions on part of documents.
XACML	Authorization	XACML (Access Control Markup Language) is a developing standard for defining Authorization Policy processing for SOAP Web Services request.	XACML defines extensions to SAML that allow complex authorization rules.
WS-Security	Confidentiality Integrity Authentication Non-Repudiation	Web Services Security is a burgeoning standard developed by major industry players that defines how to use XML-encryption and XML-Signature standards with Web Services SOAP messages.	WS-Security defines security standards, including signature.
WS-Security Extensions:	All	WS-Security extensions are being developed to add improved security functionality to the WS-security standard.	

Figure 4 – Overview of Standards under Development

4.4.4 References

1. The Internet Engineering Task Force, <http://www.ietf.org/>
2. World Wide Web Consortium (W3C), *XML Signature WG*, <http://www.w3.org/Signature/>
3. World Wide Web Consortium (W3C), *XML Encryption Syntax and Processing*, <http://www.w3.org/TR/2002/PR-xmlenc-core-20021003/>
4. WS-Federation, *Specification: Web Services Security (WS-Security)*, <http://www-106.ibm.com/developerworks/webservices/library/ws-secure>

4.5 Encryption for Web Components

4.5.1 Background

Encryption techniques are widely utilized to protect sensitive information. But implementation of encryption technologies may introduce security flaws associated with encryption into Web applications. The biggest security risk involves the compromise of storage keys, digital certificates and passwords. Poor choice of an encryption algorithm and sources of randomness (for generating keys) are also major security issues. Since encryption is primarily used to protect the most sensitive assets of a Web application, a successful attack can result in an extremely serious compromise of system or customer data.

Encryption and cryptographic analysis is a very extensive field of investigation, and its details are outside the scope of this document. The References contain resources with additional background information. The goal of this section is to provide a very brief overview of encryption only in the context of its use in protecting Web applications.

The goal of encryption is to make data unintelligible to unauthorized readers. Unencrypted data is more easily intercepted, and possibly altered. Encryption protects both the confidentiality and integrity of information. (Although encrypted data could be altered if intercepted, it could not be successfully decrypted, so modification attempts would be detected.) A robust encryption mechanism if effectively implemented will be difficult to decipher if attacked. Encryption operations are typically performed by a transformation that uses secure encryption keys. The randomness of keys makes encrypted data harder to attack. Keys also perform the function of decrypting data. Encryption is one of the most effective ways to achieve data security. In order to read an encrypted file, an individual must have access to a secret key or password that enables decryption of the data. These security components enable widespread implementation of cryptographic services in applications and the enterprise infrastructure. Digital certificates use encryption techniques to protect and authenticate entities.

The security of encrypted data depends on several factors, including the algorithm used, how the algorithm is implemented, and the key size. Section 4.2.4 – Server-Sever Trust in Web System Design describes encryption security mechanism for protecting server to server communication. Data Encryption Standard (DES) is a common algorithm that is used by many encryption protocols. Two of the most common uses for encryption in Web applications are the SSL and TLS protocols used to protect data during transmission between client and browser or between Web application components.

Secure Socket Layer (SSL)

The SSL security protocol (version 3) provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. Because SSL is built into all major browsers and Web servers, simply installing a digital certificate turns on their SSL capabilities. SSL provides support for a variety of encryption algorithms and key lengths. The most common key sizes are 40-bits and 128-

bits. These lengths refer to the "session key" used for encryption of routine transactions with the RC4 protocol. SSL also provides. The session key is initially generated and exchanged through a handshake protocol that makes use of public key-private key encryption. Several additional encryption algorithms are supported by SSL, including DES (56 bit key), triple DES (168 bit keys) and AES (128 bit keys).

Transport Layer Security (TLS)

The TLS and SSL protocols are essentially identical and provide similar functionality. Version 1 of the TLS protocol is the IETF standard based primarily on SSL v.3, and provides server and client authentication and encryption during data transit process. TLS allows a Web client to confirm a Web server's identity and vice versa. TLS uses cryptography techniques to check that a server's name and public key are contained in a valid certificate issued by a certificate authority.

4.5.2 Standard

1. All encryption algorithms and protocols used in FSA Web applications must be approved for use in Federal information systems.
2. SSL/TLS encryption should be used for external communications between Web browsers and servers when sensitive information will be exchanged. Examples of sensitive information include personal information, financial information, medical information, or any data subject to Privacy Act protections.
3. Data stored in databases should be encrypted when it is extremely sensitive, or when the data is used as security credentials to govern access to sensitive data. For example, passwords and PINS stored in a database should be generally be stored encrypted or as a message hash generated using cryptographic techniques.
4. Encryption of transmissions between Web application components should be considered for extremely sensitive data, if the components are located on separate network segments not adequately protected from message interception, or if any segment of the communications link between the components is over the public Internet.

4.5.3 Guidelines

The guidelines described here cover the following topics:

- Choosing Encryption Algorithms
- Encryption of Stored Data

4.5.3.1 Choosing Encryption Algorithms

Choosing an appropriate encryption algorithm depends on several factors that will vary with organization. Although at first glance it might appear that the strongest encryption

available should always be used, that is not always true. The higher the level of the encryption, the greater impact it will usually have on the Web server's resources and communications speed. AES 128, a newly adopted encryption standard, is an exception to this rule, because it provides higher performance and security than Triple-DES.

Factors that should be considered when selecting encryption algorithms are:

- Level of security required, based on:
 - Value of the data (to either the organization and/or other entities – the more valuable the data, the stronger the required encryption)
 - Time value of data (if data are valuable but for only a short time period [e.g., days as opposed to years] then a weaker encryption algorithm can be used – for example, passwords that are changed daily because the encryption needs to protect the password for only a 24-hour period)
 - Threat to data (the higher the threat level, the stronger the required encryption)
 - Other protective measures that are in place and that may reduce the need for stronger encryption. For example, using protected methods of communications such as dedicated circuits as opposed to the public Internet.
- Required performance – higher performance requirements may require procurement of additional system resources such as a hardware cryptographic accelerators or necessitate use of encryption protocols with lower performance demands. Major considerations include:
 - System resources (less resources [e.g., process, memory] may necessitate weaker encryption)
 - Import, export, or usage restrictions
 - Encryption schemes supported by Web server application
 - Encryption schemes supported by Web browsers of expected users.

The table below, from NIST SP800-44, summarizes additional information on selection of encryption algorithms, referred to as “cipher suites” in IETF standards.

Recommended Use	Cipher Suites
Highest Security	Encryption: Advanced Encryption Standard (AES) 256-bit encryption Authentication & Digest: Digital Signature Standard, (DSS) or RSA with 2048 bit keys, and Secure Hash Algorithm-1 (SHA-1)
Security and Performance	Encryption: AES 128-bit encryption Authentication & Digest: DSS or RSA with 1024-bit keys, and SHA-1
Security and Compatibility	Encryption: AES 128-bit encryption with fallback to Triple Data Encryption Standard (3DES) 168/112-bit encryption (note: 3DES is considerably slower than AES) Authentication & Digest: DSS or RSA with 1024-bit keys, and SHA-1
Authentication and Tamper Detection	Authentication & Digest: DSS or RSA with 1024-bit keys and SHA-1

Figure 5 - Recommended uses of encryption algorithms approved for federal information systems.

4.5.3.2 Encryption of Stored Data

Data stored in databases may be encrypted to protect it from unauthorized users. In most cases, encryption of databases is reserved for only the most sensitive data, such as security credential information. Use of encryption for other types of data, such as personal information, financial data, or medical information, must be carefully weighed against the costs in terms of implementation effort, performance, and usability of the system.

Although encryption of stored data provides a high level of security, the following disadvantages should be considered:

- Encryption may increase database size
- Encryption may require resizing database columns
- Encryption usually decreases general database performance
- Encryption slows queries because of the computational overhead of decrypting table contents during searches
- Encrypted fields cannot be indexed

4.5.4 References

1. Federal Information Security Management Act (FISMA), 2002
2. NIST Special Publication 800-44, *Guidelines on Securing Public Web Servers*, 2002

3. NIST Special Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, 1999
4. Schneir, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc., 1996, Second Edition

5 Implementation Approach

The Web security standards described in this document are in Draft form. A number of additional implementation steps will be required to ensure they are adequately communicated and used by the FSA employees and contractors responsible for designing, developing, and deploying Web applications. The steps below outline proposed steps for adopting these standards.

5.1 *Review of Standards*

These standards should be reviewed by business and technical stakeholders to validate that they meet FSA requirements for protecting Web applications. The following components of the review process will need to be defined:

- Reviewers – representation should be included from the following major areas –
 - Web application designers and developers
 - Business owners
 - Data center and technical architecture groups
 - Enterprise architecture
 - External reviewers (e.g., at the Department of Education level)
- Reviewer requirements
 - Which reviews must be sequential, and which reviews may proceed in parallel
 - Which reviewers are required, and which are optional
- Communications process that define how reviews will be conducted and how comments will be collected.
- Review timeline that defines when comments will be due, how many rounds of review will be conducted, and when the review is expected to be complete.
- Final approvals that are required to formally adopt the standards.

5.2 *Publication of Standards*

Following adoption of the standards, they must be effectively published and communicated. Successful implementation of the policy will depend on its accessibility and visibility. The policy should be incorporated into the standard FSA IT Security and Privacy Policy documentation maintained by FSA. The policy should also be posted to the FSA security web site, and linked to other internal FSA information sites.

A communication program should be developed to inform FSA employees and contractors. Possible methods for communicating the standards include:

- Introductory memo – sent out to all members of the target audience, preferably from a senior executive, introducing the standard and briefly explaining the contents and their importance.

- Educational postcards – memos, summaries, or other vehicles that present an overview of the standards.
- Ongoing employee and contractor training – existing employees and contractors responsible for designing, developing, or managing Web applications should receive a briefing on the new standards.
- New employee training – new employees that will be involved in development of Web applications should receive training on the standards.

5.3 *Enforcement and Monitoring*

The Web security standards should be incorporated into the FSA SLC to provide for monitoring and enforcement. The standards may be provided as an addendum to the SLC, or used to develop checklists for use during design and development of Web applications.

The Web security standards will need to be periodically reviewed to maintain currency with changes in Web technologies, threats, and recommended security controls for Federal information systems. Once adopted, the Web security standards themselves should need only infrequent updating. However, recommendations and background information in the Guidelines section may need more frequent updating because of the rapidity with which Web technologies, industry trends, and security threats typically change. The Web security standards should be reviewed at least annually to check whether updates are required.

Appendix A: List of TCP/IP Ports to Consider Blocking

The table below list ports associated with common TCP/IP security vulnerabilities. The port should be blocked at the firewall unless specifically required for business purposes. Although these ports could all be blocked individually in firewalls and router access control lists, the recommended approach is to block all ports that don't have specific requirements to be open. Required ports can then be allowed as needed.

Name	Port	Protocol	Description
Small services	<20	tcp/udp	small services
FTP	21	tcp	file transfer
SSH	22	tcp	login service
TELNET	23	tcp	login service
SMTP	25	tcp	mail
TIME	37	tcp/udp	time synchronization
WINS	42	tcp/udp	WINS replication
DNS	53	udp	naming services
DNS zone transfers	53	tcp	naming services
DHCP server	67	tcp/udp	host configuration
DHCP client	68	tcp/udp	host configuration
TFTP	69	udp	miscellaneous
GOPHER	70	tcp	old WWW-like service
FINGER	79	tcp	miscellaneous
HTTP	80	tcp	Web
alternate HTTP port	81	tcp	Web
alternate HTTP port	88	tcp	Web (sometimes Kerberos)
LINUXCONF	98	tcp	host configuration
POP2	109	tcp	mail
POP3	110	tcp	mail
PORTMAP/RPCBIND	111	tcp/udp	RPC portmapper
NNTP	119	tcp	network news service
NTP	123	udp	time synchronization
NetBIOS	135	tcp/udp	DCE-RPC endpoint mapper
NetBIOS	137	udp	NetBIOS name service
NetBIOS	138	udp	NetBIOS datagram service
NetBIOS/SAMBA	139	tcp	file sharing & login service
IMAP	143	tcp	mail
SNMP	161	tcp/udp	miscellaneous
SNMP	162	tcp/udp	miscellaneous
XDMCP	177	udp	X display manager protocol
BGP	179	tcp	miscellaneous
FW1-secureremote	256	tcp	CheckPoint FireWall-1 mgmt
FW1-secureremote	264	tcp	CheckPoint FireWall-1 mgmt
LDAP	389	tcp/udp	naming services
HTTPS	443	tcp	Web
Windows 2000 NetBIOS	445	tcp/udp	SMB over IP (Microsoft-DS)
ISAKMP	500	udp	IPSEC Internet Key Exchange
REXEC	512	tcp	} the three
RLOGIN	513	tcp	} Berkeley r-services
RSHELL	514	tcp	} (used for remote login)

Name	Port	Protocol	Description
RWHO	513	udp	miscellaneous
SYSLOG	514	udp	miscellaneous
LPD	515	tcp	remote printing
TALK	517	udp	miscellaneous
RIP	520	udp	routing protocol
UUCP	540	tcp/udp	file transfer
HTTP RPC-EPMAP	593	tcp	HTTP DCE-RPC endpoint mapper
IPP	631	tcp	remote printing
LDAP over SSL	636	tcp	LDAP over SSL
Sun Mgmt Console	898	tcp	remote administration
SAMBA-SWAT	901	tcp	remote administration
Windows RPC programs	1025	tcp/udp	} often allocated
Windows RPC programs	To		} by DCE-RPC portmapper
Windows RPC programs	1039	tcp/udp	} on Windows hosts
SOCKS	1080	tcp	miscellaneous
LotusNotes	1352	tcp	database/groupware
MS-SQL-S	1433	tcp	database
MS-SQL-M	1434	udp	database
CITRIX	1494	tcp	remote graphical display
WINS replication	1512	tcp/udp	WINS replication
ORACLE	1521	tcp	database
NFS	2049	tcp/udp	NFS file sharing
COMPAQDIAG	2301	tcp	Compaq remote administration
COMPAQDIAG	2381	tcp	Compaq remote administration
CVS	2401	tcp	collaborative file sharing
SQUID	3128	tcp	Web cache
Global catalog LDAP	3268	tcp	Global catalog LDAP
Global catalog LDAP SSL	3269	tcp	Global catalog LDAP SSL
MYSQL	3306	tcp	database
Microsoft Term. Svc.	3389	tcp	remote graphical display
LOCKD	4045	tcp/udp	NFS file sharing
Sun Mgmt Console	5987	tcp	remote administration
PCANYWHERE	5631	tcp	remote administration
PCANYWHERE	5632	tcp/udp	remote administration
VNC	5800	tcp	remote administration
VNC	5900	tcp	remote administration
X11	6000-6255	tcp	X Windows server
FONT-SERVICE	7100	tcp	X Windows font service
alternate HTTP port	8000	tcp	Web
alternate HTTP port	8001	tcp	Web
alternate HTTP port	8002	tcp	Web
alternate HTTP port	8080	tcp	Web
alternate HTTP port	8081	tcp	Web
alternate HTTP port	8888	tcp	Web
Unix RPC programs	32770	tcp/udp	} often allocated
Unix RPC programs	To		} by RPC portmapper
Unix RPC programs	32899	tcp/udp	} on Solaris hosts
COMPAQDIAG	49400	tcp	Compaq remote administration
COMPAQDIAG	49401	tcp	Compaq remote administration
PCANYWHERE	65301	tcp	remote administration

Figure 6 - List of TCP/IP Ports Commonly Associated with Network Attacks

Appendix B: References

1. Apache-server.com, *Apache Server Resources*, <http://apache-server.com>
2. Apache-server.com, *Securing Your Web Pages with Apache*, <http://apache-server.com/tutorials/LPauth1.html>
3. The Center for Education and Research in Information Assurance and Security (CERIAS), <http://www.cerias.purdue.edu>
4. Security Network Servers, Computer Emergency Response Team (CERT/CC) *Securing Public Web Servers*, <https://www.cert.org/security-improvement/modules/m111.html>
5. Computer Emergency Response Team, <http://www.cert.org>
6. Khare, Rohit; Rifkin, Adam; *Trust Management on the World Wide Web*, http://www.firstmonday.dk/issues/issue3_6/khare
7. Federal Information Security Management Act (FISMA), 2002
8. Garfinkel, S. And Spafford, G., *Practical UNIX Security*, O'Reilly, 1991
9. Garfinkel, S. And Spafford, G., *Web Security and Commerce*, O'Reilly, 1997
10. Internet Society, *The Internet Engineering Task Force (IETF)*, <http://www.ietf.org/>
11. NIST Special Publication 800-42, *Guideline on Network Security Testing*, October 2003
12. NIST Special Publication 800-64, *Security Considerations in the Information System Development Life Cycle*, October 2003
13. NIST Special Publication 800-44, *Guidelines on Securing Public Web Servers*, September 2002
14. NIST Special Publication 800-46, *Security for Telecommuting and Broadband Communications*, August 2002
15. NIST, *Introduction to Public Key Technology and the Federal PKI Infrastructure*, February 2001
16. NIST Special Publication on Intrusion Detection System, *Intrusion Detection Systems*,
17. NIST ICAT Metabase, *Guidelines on Securing Public Web Servers*, <http://icat.nist.gov>
18. The Open Web Application Security Project (OWASP), *The Ten Most Critical Web Application Security Vulnerabilities*, January 2003
19. The Open Web Application Security Project (OWASP), *A Guide to Building Secure Web Applications*, September 2002
20. Organization for the Advancement of Structured Information Standards (OASIS), *Security Services TC*, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
21. Organization for the Advancement of Structured Information Standards (OASIS), <http://www.oasis-open.org/committees/xacml/repository/oasis-xacml-1.0.pdf>
22. Peteanu, Ravazan, razvan.peteanu@rogers.com, *Best Practices for Secure Development*, October 2001
23. Sanctum, Inc, *The Ten Most Common Application-Level Hacker Attacks*, November 2003

24. Schneir, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc., 1996, Second Edition
25. Security Administration, Networking, and Security (SANS) Institute, *The Twenty Most Critical Internet Security Vulnerabilities*, <https://www.sans.org/top20>, October 2003
26. Shockwave Security Issues, <http://www.webcomics.com/shockwave>
27. Web Services Interoperability Organization (WS-I), <http://www.ws-i.org>
28. World Wide Web Consortium (W3C), *Decryption Transform for XML Signature*, <http://www.w3.org/TR/2002/PR-xmlenc-decrypt-20021003>
29. World Wide Web Consortium (W3C), *XML Encryption WG*, <http://www.w3.org/Encryption/2001>
30. World Wide Web Consortium (W3C), *XML Encryption Syntax and Processing*, <http://www.w3.org/TR/2002/PR-xmlenc-core-20021003/>
31. World Wide Web Consortium (W3C), *XML Key Management Specification (XKMS)*, <http://www.w3.org/TR/xkms2>
32. World Wide Web Consortium (W3C), *XML Signature WG*, <http://www.w3.org/Signature/>
33. World Wide Web Consortium (W3C), *The WWW security FAQ*, <http://www.w3.org/Security/Faq/www-security-faq.html>
34. WS-Federation, *Specification: Web Services Security (WS-Security)*, <http://www-106.ibm.com/developerworks/webservices/library/ws-secure>

Appendix C: Additional Security Definitions

Assessment: An analysis of the vulnerabilities of an information system. Information acquisition and review process designed to assist an administrators to determine how best to use resources to protect information in systems.

Assurance: A measure of confidence that the security features and architecture of an information system accurately mediate and enforce the security policies.

Attack: An attempt to bypass security controls on a system or an application. The attack may alter, release, or deny data. Whether an attack will succeed depends on the vulnerability of the system and the effectiveness of existing countermeasures.

Audit Trail: In computer security systems, a chronological record of system resource usage. This includes user login, file access, other various activities, and whether any actual or attempted security violations occurred, legitimate and unauthorized.

Authentication: The process used to verify the identity of a user, device or any other entity in a computer system with a certain level of confidence. Authentication is the prerequisite for allowing access to resources on a system.

Breach: The successful defeat of security controls which could result in a penetration of the system.

Intrusion Detection: Pertaining to techniques which attempt to detect intrusion into a computer or network by observation of actions, security logs, or audit data. Detection of break-ins or attempts either manually or via software expert systems that operate on logs or other information available on the network.

Non-Repudiation: Method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data.

Outsourced solution: Services or capabilities that are provided by a third-party organization.

Penetration Testing: The portion of security testing in which the evaluators attempt to circumvent the security features of a system. The evaluators may be assumed to use all system design and implementation documentation, that may include listings of system source code, manuals, and circuit diagrams. The evaluators work under the same constraints applied to ordinary users.

Risk: Risk is future probability of the likelihood that a particular vulnerability will be exploited.

Security control: A protective measure that may consist of technical measures to secure the confidentiality of information systems.

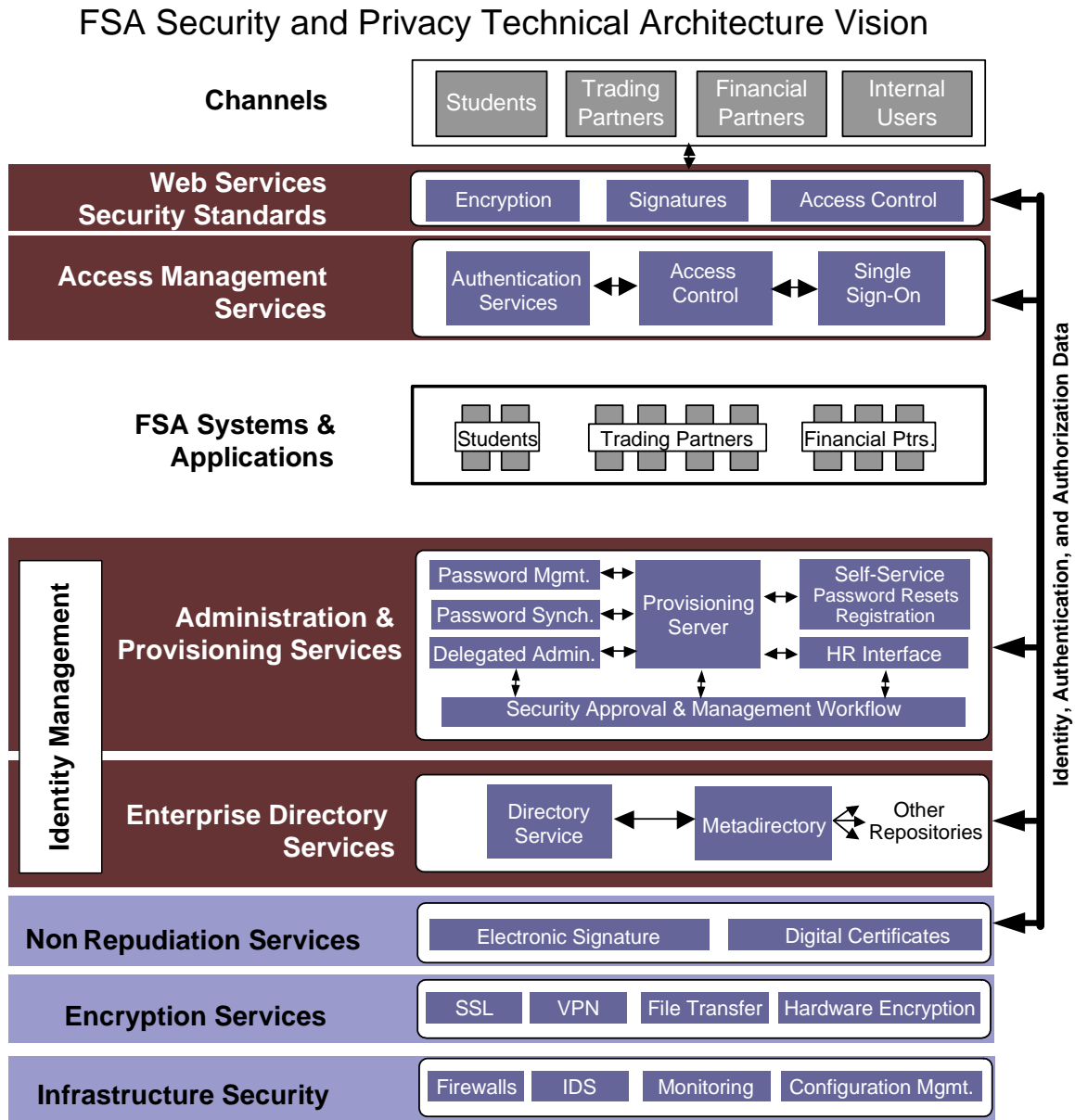
Security Architecture: A detailed description of all aspects of the system that relate to security, along with a set of principles to guide the design. A security architecture describes how the system is put together to satisfy the security requirements.

Threat: An avenue by which a vulnerability can be exploited to attack a network resource (e.g., a flood or a lightning strike is a physical threat to network resources—a local user or an Internet "hacker" is a personal threat)

Web application: A program that uses HTTP for its core communication protocols and delivers and delivers Web based information to the user.

Web application components: A Web Application contains an application's resources, such as servlets, JavaServer Pages (JSPs), JSP tag libraries, and any static resources such as HTML pages and image files.

Appendix D: FSA Security and Privacy Technical Architecture



U.S. Department of Education - Office of Federal Student Aid

Version 7-31-2003

Figure 7 – FSA Security and Privacy Technical Architecture